IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: John R. Chase	Attorney Docket No.: ALTRP098/A1185
Application No.: 10/693,546	Examiner: Lo, Suzanne.
Filed: October 23, 2003	Group: 2128
Title: METHODS AND APPARATUS FOR AUTOMATED TESTBENCH GENERATION	Confirmation No: 3624

CERTIFICATE OF EFS-WEB TRANSMISSION
I hereby certify that this correspondence is being transmitted electronically through EFS-WEB to the Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450 on June 10, 2009.

Signed: _____/Danielle Jones/_____
Danielle Jones

PREAPPEAL BRIEF REQUEST FOR REVIEW

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a Notice of Appeal.

The review is requested for the reasons stated below.

REMARKS

Claims 1-27 are pending. No claims are amended.

35 U.S.C. 101

The Examiner rejected claims 25-27 under 35 U.S.C. 101 and maintained the rejection in the advisory action. The Applicants respectfully disagree. The Examiner argues the claims are "directed to software per se. While the purpose of the software when executed may produce a tangible result, the software itself is not tangible and therefore is nonstatutory per se." Applicants submit that the claims are not directed to software per se because claims 25 is properly directed to an apparatus in a means plus function format and recites "processing means" and "interface means." These means elements provide structural and functional interrelationships between functional elements and the rest of a system such as a computing processing system, an example of which is shown at Figure 9.

Even if claims 25-27 are directed to software, Applicants respectfully submit that the subject matter of claims 25-27 is statutory. Applicants submit that the claimed invention is a practical application thereof under MPEP 2106.IV.C.2 because it is transformative and/or "otherwise produces a useful, concrete, and tangible result." The claimed invention transforms a top level module and submodules of a design module library to become a test design by "instantiating the I/O structure of a top level module," "selecting a plurality of submodules from a design module library," "parameterizing the plurality of submodules," and "providing logic to interconnect the plurality of parameterized submodules." The module and submodules represent components of electronic designs and are transformed to a different state or thing, namely test designs. The result is useful because it generates specifically and substantially a "plurality of test designs having varied characteristics to allow testing of a design automation tool." The result is tangible because real-world test designs are generated. The result is also concrete because the result is "substantially repeatable." For at least the above reasons, withdrawal of this rejection is respectfully requested.

35 U.S.C. 103

Claims 1-2, 5, 17-21, and 25-27 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten (U.S. Patent No. 5,805,795) in view of Bening et al. ("Optimizing Multiple EDA Tools within the ASIC Design Flow"). Claims 4, 6-13, 15, and 22-24 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in

further view of Zaidi (2002/0038401 A1). Claims 14-16 were rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in further view of Zaidi and Goossens ("Design of Heterogeneous ICs for Mobile and Personal Communication Systems"). Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Whitten in view of Bening and in further view of Rajsuman (U.S. Patent No. 6,678,645). Applicants respectfully traverse and address Examiner's Advisory Action below.

Whitten is directed to a "method for selecting a set of test cases which may be used to test a software program product . . . The program to be tested may have a number of code blocks that may be exercised during execution of the program. The method includes identifying each of the code blocks that may be exercised, and determining a time for executing each of the test cases in the set. A set of the test cases is then selected that exercises a maximum number of the identified code blocks that can be exercised in a minimum time." (Abstract) Applicants refer to the Background section of Whitten quoted in a previous response that discusses how a "suitable test for a particular software product" is developed. The quoted passage describes a software test designer "generat[ing] hundreds, even thousands, of test cases for a typical complex software product" to test "the validity of the assertions" and that it is difficult to "choos[e] a particular set of test cases to apply ... particularly if an optimal ... set ... is sought." Together with the Abstract, it is clear that the Whitten method seeks to produce this optimal "set of test cases" from the "hundreds, even thousands, of test cases" generated by a software test designer. Note that Whitten's code blocks are a part of the program to be tested. Each test case is separately correlated to a number of code blocks it tests. Whitten is directed to method for selecting an optimal set of test cases from existing cases.

The Examiner relies on Whitten to describe "generating a plurality of test designs." The Examiner cites column 8, lines 15-21 as support for "generating a plurality of test designs." The passage states:

> After the iterative process has been completed, a string is derived for use as the final population which has the highest fitness value. This string is interpreted, as described above with reference to FIG. 2. Once the set of test cases has been determined, it may be used to achieve a good tradeoff between test coverage and test execution time on any target operating system and hardware.
> passage states:

Applicants submit that this passage does not support "generating a plurality of test designs." Whitten discloses selecting and determining a set of test cases to apply for testing from the

"hundreds, even thousands, of test cases" created by a software test designer. Determining "the set of test cases" in this context is not "generating a plurality of test designs" in the claimed invention because the "set of test cases" are determined and/or selected from a collection of existing test cases and the "plurality of test designs" is generated by performing a number of operations as recited for each of the plurality of test designs.

In the Advisory Action, the Examiner restated the argument above as "test designs are not test cases." While Whitten's test cases and claimed test designs are both used to test a software product, there are remarkable differences. The test designs used to test a design automation tool are electronic designs that can be implemented on an electronic device (page 3, lines 4-7). Further, the claim element at issue is "generating a plurality of test designs." Whitten is silent on how to generate its test cases. Even if Whitten discloses how to generate a test case, such disclosure would not apply to the operations for generating a test design as claimed. For example, a test case is correlated to different code blocks in the software program to determine an optimal set of test cases (Whitten, column 1, lines 34-47), but a test design is not correlated to any part of the design automation tool in forming the test bench (specification, page 3, lines 4-22).

The Examiner also relies on Whitten to describe "selecting a plurality of submodules from a design module library" and "wherein a probabilistic function is applied to select submodules of different types from the library." The Examiner responded to Applicants' previous argument that "the blocks of code executed by the test cases, as recited in column 5, lines 22-52, are submodules." Applicants respectfully disagree.

Whitten's blocks of code are references to components of the software product being exercised or tested, for example, "subroutines, calls, branches, or the like" (Whitten, column 4, line 61). On the other hand, submodules, as disclosed in the specification, are parts of the test design, not parts of the design automation tool being tested (specification, page 3, lines 14-22). The relationship between submodules and other components of a test design is described in the specification at page 3, lines 14-22. Together, submodules and other components of a test design form an electronic design that can be implemented on an electronic device through a design automation tool. "Possible submodules include memory modules 431 and 433, phase locked loop module 451, adder 453, filter 455, and timer 457. The top-level module 401 also includes registers 441, 443, 445, and 447. In this example, other submodules included in the top-level module 401 are DSP core 461, processor core 463, etc. Each submodule may also be associated

with various input and output lines. The various input and output lines can be categorized as clock lines, control lines, or data lines." (page 12, lines 25-31). These submodules are design modules selected "from a design module library" to generate "a plurality of test designs." These submodules combine to form the test design

The "code blocks" tested by the test cases of Whitten are not submodules. These code blocks are a part of the product program to be tested. Column 5, lines 22-52 of Whitten describes using a compiler to identify "all of the code blocks in the program 12 to be tested " by "generat[ing] a list 16 of code blocks contained in the product source code as well as an instrumented version of the product." Thus, the code blocks are parts of the product source code such as "subroutines, calls, branches, or the like."

In the Advisory Action, the Examiner "notes that the specification describes possible submodules but does not define the necessary characteristics of a submodule." Applicants submit that the specification makes clear through its discussion the "necessary characteristics of a submodule." "Each submodule may also be associated with various input and output lines. The various input and output lines can be categorized as clock lines, control lines, or data lines" (page 12, lines 29-31). "In many real world examples, the various input lines of the top-level module and the submodules would be interconnected with design functionality to allow performance of particular tasks" (page 13, lines 1-3). Applicants submit that an explicit definition of "the necessary characteristics of a submodule" in the specification is not required because one skilled in the art would recognize these characteristics through the discussion in the specification and possible submodules cited in the previous response, such as memory, phase locked loop, adder, filter, timer, DSP core, and processor core.

Furthermore, these code blocks are not "selected" "from a design module library." The Examiner may have associated the collection of test cases "initially generated or collected by a test designer" (Whitten, column 4, line 65-66) to a "design module library." However, this association is incorrect because test cases are not submodules. The example submodules from the specification include memory modules, phase locked loop module, adder, filter, timer, registers, DSP core, and processor core. (page 12, lines 25-31) Test cases designed to test a software program product are not any of these. In the Advisory Action, the Examiner points to Whitten's column 5, lines 23-52 wherein "Whitten maintains a list of code blocks to be selected and are used to generate test cases." While the cited passage does describe a list of code blocks, the list is used to "identify the optimum set of test cases" (column 5, line 23), to "determine[e]

the extent or coverage that a particular set of test cases provides" (column 5, lines 36-37), to "derive a subset of the test cases that exercises a maximum number of the identified code blocks" (column 5, lines 45-46), and not to "generate test cases."

Bening does not cure the defects of Whitten. Bening discloses a "proposed design methodology" to "let engineers optimize a design's functionality and simultaneously support multiple EDA tools." (Abstract, page 46) Bening is about a design methodology and not about EDA tool testing or generating test designs. The Examiner points to various portions of Bening as disclosing various claim elements that are not supported by Bening. For example, the Examiner points to Bening at page 47 as disclosing "creating a test design for an EDA tool comprising instantiating the I/O structure of a top level module." The cited passage does not teach "instantiating the I/O structure of a top level module." Instead, it teaches that "Verilog can link a different definition for a specific, instantiated module by referencing application-oriented libraries." Applicants believe that linking "a different definition for a specific, instantiated module" does not teach one skilled in the art to "instantiating the I/O structure of a top level module." The Advisory Action argues that "one must first instantiate the instantiated module" in order to "link a definition." While this statement may be true, Bening does not disclose information to teach one skilled in the art to "instantiat[e] the I/O structure of a top level module," even if it must be performed first.

CONCLUSION

In light of the above remarks, the rejections to the independent claims are believed overcome for at least the reasons noted above. Applicants believe that all pending claims are allowable in their present form. Please feel free to contact the undersigned at the number provided below if there are any questions, concerns, or remaining issues.

Respectfully submitted,
Weaver Austin Villeneuve & Sampson LLP

/Cindy H. Shu/
Cindy H. Shu
Reg. No. 48,721

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100

APPENDIX: PENDING CLAIMS

1. (Previously Presented) A method, comprising:

generating a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein generating one of the plurality of test designs comprises:

instantiating the I/O structure of a top level module, the top level module having input and output pins;

selecting a plurality of submodules from a design module library, wherein a probabilistic function is applied to select submodules of different types from the library;

parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

applying the plurality of test designs to test the design automation tool.

2. (Previously presented) The method of claim 1, wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

3. (Previously presented) The method of claim 1, wherein the plurality of submodules comprise a memory module and a Digital Signal Processor (DSP) core.

4. (Previously presented) The method of claim 1, wherein instantiation constraints are used to select the plurality of submodules.

5. (Previously presented) The method of claim 1, wherein the design automation tool is a synthesis or a place and route tool.

6. (Previously presented) The method of claim 1, wherein providing logic to interconnect the plurality of parameterized modules comprises identifying inputs and outputs.

7. (Previously presented) The method of claim 6, wherein inputs comprise input pins of the top level module, submodule output lines, and registers.

8. (Previously presented) The method of claim 6, wherein outputs comprise output pins of the top level module, submodule input lines, and registers.

9. (Previously presented) The method of claim 8, wherein providing logic to interconnect the plurality of parameterized modules further comprises classifying inputs and outputs as clock lines, control lines, and data lines.

10. (Previously presented) The method of claim 8, wherein generating one of the plurality of test designs further comprises:

generating randomized logic.

11. (Previously presented The method of claim 10, wherein randomized logic is generated to drive outputs.

12. (Previously presented) The method of claim 10, wherein generating randomized logic comprises directly wiring outputs to inputs, generating a logic expression using inputs, generating a mathematical expression using inputs, or generating decision logic.

13. (Previously presented) The method of claim 6, wherein parameterizing the plurality of submodules comprises defining interfaces, data width, and the type of signal for input and output lines associated with the submodule.

14. (Previously presented) The method of claim 6, wherein submodules comprise adders, phase lock loops, memory, and timers.

15. (Previously presented) The method of claim 6, wherein generating one of the plurality of test design further comprises selecting a clock structure for each output.

16. (Previously presented) The method of claim 15, wherein clock structures include a plurality of synchronous and asynchronous structures.

17. (Previously Presented) A computer system, comprising:

memory operable to hold information associated with a design module library;

a processor coupled to memory, the processor configured to generate a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein generating one of the plurality of test designs comprises:

instantiating the I/O structure of a top level module, the top level module having input and output pins;

selecting a plurality of submodules from the ~~a~~ design module library, wherein a submodules of different types are randomly selected from the library;

parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

applying the plurality of test designs to test the design automation tool.

18. (Original) The computer system of claim 17,wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

19. (Original) The computer system of claim 17,wherein the design automation tool is used to implement designs on an ASIC.

20. (Original) The computer system of claim 17,wherein the design automation tool is an electronic design automation tool.

21. (Original) The computer system of claim 17,wherein the design automation tool is a synthesis or a place and route tool.

22. (Original) The computer system of claim 17, wherein providing logic to interconnect the plurality of parameterized modules comprises identifying inputs and outputs.

23. (Original) The computer system of claim 22, wherein inputs comprise input pins of the top level module, submodule output lines, and registers.

24. (Original) The computer system of claim 22, wherein outputs comprise output pins of the top level module, submodule input lines, and registers.

25. (Previously Presented) An apparatus for generating test a testbench, the apparatus comprising:

processing means for generating a plurality of test designs, the plurality of test designs having varied characteristics to allow testing of a design automation tool, wherein means for generating one of the plurality of test designs comprises:

means for instantiating the I/O structure of a top level module, the top level module having input and output pins;

means for selecting a plurality of submodules from a design module library, wherein a probabilistic function is applied to select submodules of different types from the library;

means for parameterizing the plurality of submodules from the design module library for interconnection with the top level module, the plurality of submodules having input and output lines;

means for providing logic to interconnect the plurality of parameterized submodules as well as to connect the plurality of parameterized submodules to various input and output pins of the top level module;

interface means for applying the plurality of test designs to test the design automation tool.

26. (Previously presented) The apparatus of claim 25, wherein the design automation tool is used to implement hardware descriptor language designs on a programmable chip.

27. (Previously presented) The apparatus claim 25, wherein the design automation tool is used to implement designs on an ASIC.